VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza obrazu a detekce objektů pro hru fotbal robotů

Image Analysis and Object Recognition for Robot Soccer

2015 Dominik Dolkoš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dominik Dolkoš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Analýza obrazu a detekce objektů pro hru fotbal robotů
Image Analysis and Object Recognition for Robot Soccer

Zásady pro vypracování:

V bakalářské práci se student zaměří na modul analýzy obrazu, jež je součástí vznikající knihovny pro fotbal robotů. Student bude v rámci práce spolupracovat s kolegy, jež se budou zabývat detekcí vzorů chování pro pohyb robota s cílem vytvořit komplexní systém pro hru fotbalu robotů.

Cíle práce:
1. Prostudovat metody analýzy obrazu použitelné pro detekci objektů na hřišti.
2. Optimalizace stávající metody analýzy obrazu, případně nahrazení za vhodnější metodu.
3. Ve výsledku bude vytvořen modul analýzy obrazu, použitelný v knihovně pro hru reálných robotů.

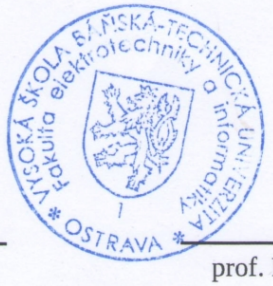
Seznam doporučené odborné literatury:

[1[T. Bandlow, M. Klupsch, R. Hanek and T. Schmitt: Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario, Lecture Notes in Computer Science, 2000
[2] P.J. Thomas, R.J. Stonier, P.J. Wolfs: Robustness of colour detection for robot soccer, ICARCV, 2002
[3] J. Bruce, T. Balch, M. Veloso: Fast and inexpensive color image segmentation for interactive robots, Intelligent Robots and Systems, 2000

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Václav Svatoň**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 7.5.2015 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7.5.2015 .....

Abstrakt

Tato práce se zabývá analýzou obrazu fotbalu robotů. Jsou zde nastíněny základní metody pro zpracování obrazu a vytvořen nový postup pro získání hledaných informací. Toho je dosaženo díky použití detekce hran a jejich ztenčení. Následuje nově vytvořený algoritmus pro porozumění dat na obrázku, který využívá vlastností jednotlivých objetků pro zjištění informací o nich. Zpracování snímků probíhá ve většině případů v reálném čase. Je tak vytvořen nový modul vznikající knihovny obsluhující fotbal robotů.

Klíčová slova: bakalářská práce, fotbal robotů, analýza obrazu, detekce hran, ztenčovaní hran, porozumění obrazu, segmentace

Abstract

This thesis deals with a problem of analyzing image in robot soccer. We showcase few basic methods of image processing and we create new algorithm for detecting specific informations about the field and players. This is done by using simple edge detection method with thinning of the edges afterwards. The algorithm for image understanding is using features of individual objects to find out their whereabouts. The image processing runs mostly in realtime. Final product is new module for newly formed library for robot soccer.

Keywords: bachelor's thesis, robot soccer, image analysis, edge detection, thinning, image understanding, image segmentation

Obsah

Seznam obrázků

1 Úvod

Technologie počítačů a robotiky jdou neskutečně rychle dopředu a s tím rostou možnosti nejen vědců, ale také nadšenců a laiků. Za posledních 20 let ušla věda velký kus cesty, a proto se dnes v některých oblastech spoléhá na roboty i s velmi důležitými úkoly. Od chirurgie přes těžké obráběcí stroje až po vysavače a sekačky na trávu, které se dokaží samy navigovat. Automatické přístroje jsou dnes všude kolem nás. Existují však také odvětví, kde nadšenci z řad odborné společnosti sestrojují roboty, jejichž úkolem je hrát fotbal. Pořádají se také turnaje, na kterých vyhraje tým, který nejlépe sestrojí a naprogramuje chování jak jednotlivých robotů, tak celého týmu.

Každoročně se v různých městech světa schází stovky týmů, kde jeden sestává až z 10 lidí. Soutěží mezi sebou ve vlastním mistrovství ve sportu, v němž jsou důležitější zkušenosti, schopnost inovace a inteligence, než fyzický fond účastníků. I přesto, že soupeří proti sobě, všichni tito odborníci mají společný cíl, kvůli kterému toto odvětví robotiky vzniklo. Chtějí vytvořit tým, který bude konkurenceschopný i proti těm nejlepším z řad fotbalistů.

Proces sestavení týmu pro fotbal robotů je zdlouhavý a obnáší zapojení vědců z několika různých oborů. Jedním z nich je také zpracování obrazu buď toho, který dostáváme od samotných robotů nebo kamer umístěných nad hřištěm. Během posledních let vznikají rozsáhlé otevřené knihovny pro práci s obrazem, ve kterých své znalosti a poznatky aplikují nadšenci z celého světa.

Cílem této práce je vytvořit jednu z částí takové knihovny. Tento modul se zabývá analýzou obrazu, získaného z kamery umístěné nad hřištěm. Výstupem metod této knihovny by mělo být jak umístění hráčů a míče na hřišti, tak rozměry samotného hřiště a branek.

V první části práce je nahlédnuto do historie fotbalu robotů. Jsou zde popsány kategorie tohoto sportu a také základní idea. Poté si představíme základní metody, které jsou dnes i v minulosti využívány týmy při zpracování obrazu. Následně je navržen a implementován nový postup, který kombinuje výhody i nevýhody různých typů metod. Nakonec shrneme výsledky našeho postupu a srovnáme jej s již používanými.

Pro správné porozumění této práci je třeba základní znalosti z oblasti matematiky, počítačové grafiky a programování.

2 Historie a současnost

Fotbal robotů je v robotice již známé téma. První zmínky a nápady jsou z roku 1992 a vůbec první oficiální turnaj týmů, tzv. RoboCup [1], se uskutečnil v srpnu roku 1997, kdy se v Japonské Nagoyi sešlo přes 40 týmů z celkem 11 zemí. Původně byli rozděleni do 3 kategorií, které zahrnovaly malou a střední velikost a počítačové simulátory. Přeposledního ročníku RoboCupu (rok 2013), který se konal v Nizozemí, se zůčastnilo 410 týmů ze 45 zemí, kdy krom hlavního turnaje, jsou pořádány otevřené turnaje po celém světě. Soutěží se v 6 kategoriích:

- **2D simulace** Počítačová simulace ve dvourozměrném prostoru. O analýzu obrazu a jeho zpracování se stará společná stanice, která týmům pouze odesílá informace o situaci na hřišti (pozice hráčů a míče apod.). Programátorský úkol týmů je velmi rychle vyhodnocení situace a výběr ideálního postupu.

- **3D simulace** Taktéž počítačová simulace, ovšem narozdíl od výše zmíněné 2D simulace, jsou zde použity stejné modely robotů, které poté soutěží ve standardizované lize. Týmy tak mají možnost vyzkoušet si inovace na tomto simulátoru, než jej implementují do svých strojů.

- **Malá velikost (Small size)** Hra reálných robotů, na malém hřišti, které snímají dvě kamery umístěné 4m nad hřištěm. Informace z kamer si zpracovávají týmy na počítačích mimo hrací plochu a poté bezdrátově odesílají pokyny svým robotům. Hlavní výpočetní zátěž je tak na těchto stanicích, které vykonávají většinu analýz a výpočtů.

- **Střední velikost (Middle size)** Roboti jsou větší a už každý má svůj vlastní vestavěný počítač a kamery, takže si každý robot analyzuje hru sám. Jediná dostupná komunikace je pouze vzájemná mezi roboty, kde probíhá spolupráce robotů a také mohou přijmat pokyny od rozhodčího.

- **Využití standardizovaného prostředí (Standard platform)** Pro tuto soutěž jsou současnosti využíváni humanoidní roboti střední velikosti - H25 NAO sestaveni v laboratořích Aldebaran Robotics [2]. Všechny týmy používají stejné roboty a v této kategorii je největší snaha přiblížit podmínky reálnému fotbalu, kde mají branky stejnou barvu, hřiště je větší a hráčů je více, proto spolupráce mezi roboty musí být komplexnější.

- **Humanoid** Zaměřuje se na to, aby roboti byli co nejvíce podobní lidem a dělí se na další 3 ligy, podle velikosti robota, kde tzv. KidSize (velikost dítěte) jsou velmi

pohybliví a dynamičtí roboti velikosti 30-60 cm. TeenSize mají 90-120 cm a poté jsou zde AdultSize roboti, kteří jsou velikostně srovnatelní s člověkem menšího vzrůstu. V této poslední kategorii se jedná zatím pouze o 2 roboty, kdy jeden střílí na branku a druhý se snaží střelu chytit a poté se role obrací.

Základní myšlenkou a výzvou pro celý RoboCup [1] je do poloviny tohoto století sestavit tým robotů, který dokáže porazit vítěze Mistrovství světa v klasickém fotbale. Dnes se zdá tento úkol nemožný, ale v posledních letech urazila robotika velký kus cesty. Lidé se vyvíjí mnohonásobně pomaleji než roboti a sestavit dokonale sehraný tým robotů může být nakonec jednodušší než se zdá. Už v současnosti se každým rokem na velkých RoboCup turnajích pořádají exhibiční zápasy robotů proti lidem, ikdyž zatím to je spíše pro pobavení.

2.1 Analýza obrazu

Jedná se o klíčovou složku výpočetních procesů ve fotbale robotů. Cílem analýzy obrazu je získat co nejpřesnější rozbor aktuální situace na hřišti. Především pozice všech robotů, jak jsou natočeni, kde se nachází míč, kde jsou spoluhráčí a na které straně je soupeřova branka.

V historii musely týmy přicházet s velmi dobře optimalizovanými algoritmy, což většinou znamenalo, že byly používané metody jednodušší na úkor přesnosti a kvality rozdělení obrazu. U analýzy v reálném čase je nejdůležitější rychlost s jakou přichází výsledky. Fotbal je velmi dynamická hra, a proto není možné pracovat s odezvami v řádech sekund. Dnes máme velmi výkonné počítače, díky tomu si můžeme dovolit používat sofistikovanější algoritmy, které vykazují přesné výsledky a na starších počítačích by znamenaly příliš velké zdržení. Lze proto pozorovat přesun v pohledu na optimální přístup k analýze obrazu.

V minulosti byla nutnost pracovat na základě kontrastních barev a získavat informace z barevných složek obrazu (thresholding, region-growing metody apod.), kdežto dnes se komunita přiklání spíše k rozpoznávání okrajů objektů a jejich tvarů. Tyto algoritmy jsou náročnější, ale jsou také méně závislé na vnějších faktorech jako je osvětlení hrací plochy, nestandardní barevná označení apod.

3 Základní používané metody

Největší a časově nejnáročnější částí analýzy obrazu je jeho segmentace. Ta má za cíl rozdělit obraz na části, které mají velkou podobnost s objekty nebo oblastmi z reálného světa. Segmentaci je možné rozdělit na dva druhy. Při *Kompletní segmentaci* korespondují výsledné části s objekty, které obraz obsahuje. *Částečná segmentace* naopak znamená, že části, které jsme pomocí ní získali, nepředstavují reálné objekty úplně přesně. Ikdyž se výpočetní technika rychle vyvíjí a dává nám nové možnosti, základní algoritmy pro segmentaci zůstávají stále relevantní a důležité.

Metody segmentace se dělí do 3 skupin, dle toho, jaké vlastnosti obrazu a objektů využívají: První je segmentace, která zohledňuje celkové znalosti o obraze nebo jeho části, což může být vyjádřeno například histogramem některé z vlastností (jas, odstín barvy apod.). Druhou skupinu představují metody založené na hranách objektů, a třetí využívá přímé rozdělení obrazu na regiony.

Pro analýzu je také důležitý výběr barevného modelu a je to jeden z prvních kroků který musíme zvážit, pokud se vydáme cestou segmentace pomocí barev. Od kdysi nejčastěji používaného klasického RGB se v posledních letech upouští ve prospěch pro roboty výhodnějších reprezentací [3], jako jsou YUV, YCbCr nebo HSV. Například model HSV (Hue Intensity Value nebo také HSB - Hue Saturation Brightness), je velmi blízký tomu, jak barvy vnímají obyčejní lidé. Asi málokdo vidí barvu a řekne, že to je barva která je z 50% červená, z 30% zelená a 10% modrá. Většinou uslyšíte, že barva je výrazná (value) modrá (hue) s trochou šedé (saturation). Kdy 0% saturation a 100% value, dá vždy bílou a 0% value dá vždy černou barvu. Value se dá chápat jako příměs černé, Hue je druh barvy a Saturation je intenzita. Barvy a jejich model je jsou nutou součástí první metody.

3.1 Prahování (thresholding)

Prahování na úrovni odstínů šedé je nejjednodušší metoda segmentace. Většina objektů je charakteristická svým konstatním jasem, díky tomu je můžeme pomocí dané konstanty neboli prahu oddělit od pozadí obrazu. Prahování je nejstarší metodou segmentace. Je rychlá a výpočetně nenáročná a proto je stále široce využívána v jednoduchých aplikacích.

Kompletní segmentace obrazu je konečná množina regionů $R_1, \ldots, R_s$,

$$R = \bigcup_{i=1}^{S} R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j \tag{3.1}$$

Pomocí prahování lze dosáhnout kompletní segmentace pouze na jednoduchých obrazech. Princip segmentace pomocí prahování spočívá v rozdělení každé složky barevného

prostoru na několik částí. Ty jsou odděleny hranicemi (tzv. práhy), které jsou určeny předem, pokud je to pro danou situaci možné, nebo jsou určeny po prvotní analýze obrazu a jeho barev. Jen velmi zřídka se podaří rozdělit obraz pouze pomocí jednoho prahu.

První možnost je efektivní pouze v případě, že dopředu víme jaké důležité barvy se budou na obrázcích vyskytovat. Například u standardizovaného barevného značení ve Small Size lize RoboCupu je míček oranžový, hřiště zelené (v našem případě černé) a roboti jsou podle barev rozděleni do týmů a svých pozic. Je proto možné vždy podle potřeby upravit hodnoty, které budou sloužit jako práhy, přímo na místě.

Druhá varianta umožňuje systému se "naučit", jaké hodnoty použít pro práhy. Algoritmus "popularity"na základě barev v obraze vybere N nejčastějších výskytů barvy. To je možné udělat například pomocí histogramu každé barevné složky. Histogram je graf četnosti jednotlivých odstínů, z něhož se dá určit N. Každá barevná složka se rozdělí například na 6 částí, což vytvoří celkem 6*6*6 neboli 216 přihrádek. Z nich se vybere N barev, které budou sloužit jako základní. Každému pixelu je poté přiřazena nejbližší základní barva. Segmentace obrazu je poté kvalitnější a přesnější, ale z pravidla vyžaduje další zpracování, kvůli většímu počtu barevných oblastí.

Do nich se poté rozřazují jednotlivé pixely obrazu. Cílem tohoto postupu segmentace je získat obrázek, který bude mít pouze několik, přesně odlišitelných barev. Tato úprava zajistí přesnější a rychlejší další kroky analýzy, protože omezí rozsah a množství barev v obrázku.

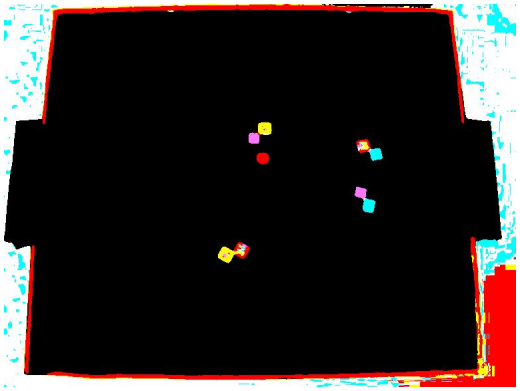
Jak je uvedeno v [4], samotný algoritmus pro rozdělení pixelů do těchto oblastí je v základu velmi jednoduchý. Pro každý pixel je nutno, v případě trojsložkových barevných modelů, provést 6 porovnání hodnoty barevné složky s hodnotou práhu pro tuto složku a to vše pro každou předem určenou "přihrádku". Například máme-li zelené hřiště, které je uměle osvětleno. Toto osvětlení vytváří na ploše různé odstíny zelené, což by mohlo, například při následné detekci hran, mít za následek mnoho falešných nálezů. Dobrý práhovací algoritmus zajistí, že hřiště bude mít pouze jeden odstín a tím se značně zjednoduší a zefektivní hledání objektů a jejich hran.

V roce 2009 byla uvedena do Small Size ligy Robocupu open-source knihovna SSL-Vision [5]. Tato knihovna by měla sloužit jako základ pro analýzu obrazu týmů a dnes je využívána řadou z nich. Tato knihovna používá pro segmetnaci obrazu právě barevné prahování a je velmi rychlá a nenáročná.

3.2 Detekce hran (edge detection)

V současnosti asi nejvýznamější metodou využívanou nejen ve Small size lize fotbalu robotů, ale v celé oblasti počítačového vidění, je detekce hran. Jak je uvedeno v [6] hranami

(a) Originální obrázek



(b) Obrazek po thresholdingu

Obrázek 1: Příklad segmentace pomocí prahování

rozumíme místa v digitálním obraze, ve kterých můžeme pozorovat změny v odstínu šedi (barevné obrázky se často pro zjednodušení převádí do odstínů šedé). Čím větší změna je, tím je jednodušší detekovat hranu. Tato místa po jejich spojení dají v ideálním případě přesné okraje, o šířce 1px, objektů na obraze. To poté umožní o poznání snadněji zjistit polohu, natočení, tvar, počet nebo například jestli se objekty překrývají. V praxi bývá výsledkem této metody obrázek kde jsou 2 kontrastními barvami označeny hrany mezi objekty a pozadím.

Detekce hran většinou není triviální metoda, jelikož ji ovlivňuje hned několik faktorů. Ty plynou z digitalizace obrázků. Prvním problém nastává u kamer, které pro účely fotbalu robotů nedosahují kvalit velkých kamerových systémů. Při převodu obrazu na digitální se například některé barevné přechody mohou zobrazit jako jakési schodiště, odstupňované od jedné barvy k druhé. Hrana poté musí být zvolena uprostřed tohoto přechodu. Druhým velmi častým problémem je šum. Tento jev způsobuje, že dva sousední pixely, které v reálu mají naprosto totožnou barvu, mohou na fotografii mít jiný odstín šedé. Rozlišujeme 2 typy šumu převládající v oblasti analýzy obrazu [6]:

- Šum nezávislý na signálu
- Šum závislý na signálu

Šum nezávislý na signálu (tzv. bílý šum) je sada různě šedých pixelů, rozmístěných na obrázku nezávisle na jeho datech. Vzniká při elektronickém přenosu z jednoho místa na druhé. Tento typ není složité uměle vytvořit pro potřeby testování [6], ovšem pro naše účely nejsou tyto experimenty příliš důležité, vzhledem k faktu, že pracujeme ve velmi specifických podmínkách. Není také velký problém se s tímto šumem vypořádat. Narozdíl

od šumu nezávislého, je těžší pracovat přímo s obrázkem který má velké množství *šumu závislého na signálu*. Přikladem je zrnitost některých fotografií, kde je třeba dalších úprav pro odstranění nebo zmírnění šumu.

V detekci hran rozlišujeme 3 hlavní typy operátorů pro detekci hran:

- Derivační operátor navržen pro účely nalezení míst s velkými změnami v intenzitě šedé.
- Druhý typ využívá malých vzorů (tzv. konvolučních matic) jako modelů pro hranu.
- Poslední a nejkomplexnější typ je založen na matematickém modelu hrany. Nejlepší z těchto operátorů využívají také model šumu a snaží se ho vzít v potaz.

V této práci se budeme zabývat převážně druhým typem operátorů, protože operátory prvního a třetího typu využívají výpočetně náročné matematické operace a to je pro analýzu obrazu v reálném čase nevyhovující. Konkrétně si představíme Sobelův hranový detektor [7]. Uvedeme si také jednoho zástupce, který nezapadá úplně do jedné z těchto kategorií - Cannyho hranový detektor [8] z důvodu jeho názornosti a důležitosti.

3.2.1 Sobelův hranový detektor

Je mnoho detektorů které využívají nějaký vzor pro nalezení hran objektů. Jeden z nejznámějších je Sobel edge detector, který využívá 2 konvoluční masky:

$$S_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad S_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

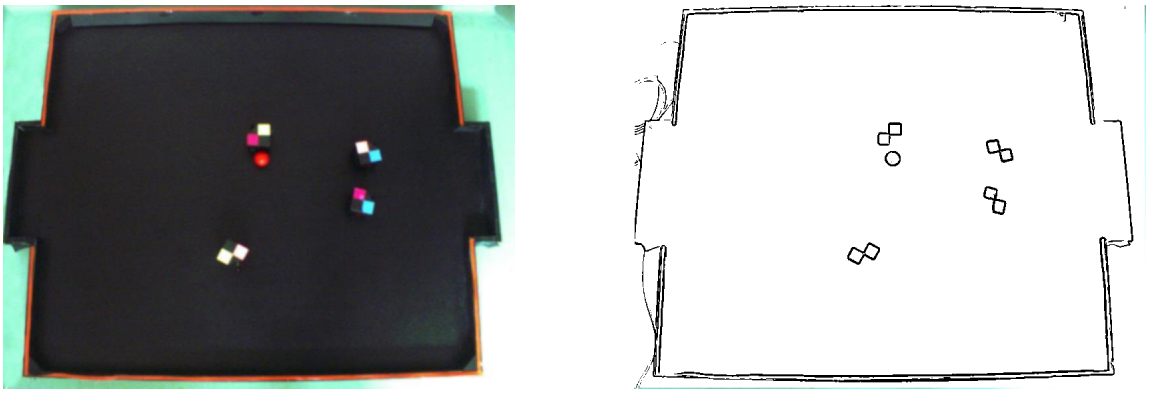
S_x představuje komponentu x a S_y je komponenta y Sobel operátoru. Obě tyto masky se aplikují na každý pixel obrázku. Například máme obrázek I v odstínech šedi. Pro pixel $I(i, j)$, kde I je intenzita šedé můžeme S_x a S_y spočítat takto ($I[i][j]$ znamená intenzita pixelu na souřadnicích (i, j)):

$$S_x = I[i-1][j+1]+2*I[i][j+1]+I[i+1][j+1]-I[i-1][j-1]-2*I[i][j-1]-I[i+1][j-1] \quad (3.2)$$

$$S_y = I[i+1][j+1]+2*I[i+1][j]+I[i+1][j-1]-I[i-1][j+1]-2*I[i-1][j]-I[i-1][j-1] \quad (3.3)$$

Poté co jsou spočítány hodnoty S_x a S_y pro každý pixel, musíme provést porovnání součtu jejich absolutních hodnot ($M = |S_x| + |S_y|$) s hodnotou, která určí zda se jedná o hranu nebo ne. To proto že všechny pixely nám po výpočtech dají nějakou hodnotu S_x a S_y a pouze ty velké značí, že se jedná o hranu.

Pokud chceme získat hrany barevného obrázku máme dvě možnosti. Buď ho převedeme do odstínu šedi nebo provedeme výpočet S_x a S_y pro každou barevnou složku a pokud je M alespoň jedné ze složek nad předem určeným prahem, pak se jedná o hranu.



(a) Originální obrázek

(b) po aplikaci Sobel edge detectoru

Obrázek 2: Příklad detekce hran pomocí Sobel edge detectoru

3.2.2 Cannyho hranový detektor

V roce 1986 si John F. Canny [8] stanovil 3 požadavky, které by měl splňovat detektor hran a popsal optimální metody k jejich dosažení. Tyto požadavky se dají shrnout jako:

- **Minimální chybovost** - Detektor hran by měl reagovat pouze na hrany a měl by je najít všechny a zároveň nedetekovat místa, která hranami nejsou.
- **Lokalizace** - Vzdálenost mezi pixely hrany, které detektor označil a reálnou hranou by měla být co nejmenší.
- **Jednoznačnost** - Detektor nesmí označit více pixelů jako hranu na místě, kde existuje pouze jeden (nesmí docházet ke zdvojení hrany).

Při zohlednění těchto 3 základních problémů se snažil Canny najít matematickou funkci, která bude maximalizovat efekty prvních dvou požadavků, při zohlednění třetího.

Přestože je tento problém velmi složité řešit analyticky, výsledek je velmi podobný *první derivaci Gaussovy funkce*. Pro připomenutí Gaussova funkce vypadá takto:

$$f(x) = e^{-\frac{x^2}{2\sigma^2}} \tag{3.4}$$

První derivace podle x je poté

$$f'(x) = \left(-\frac{x}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}} \tag{3.5}$$

která je aproximací Cannyho optimálního filteru pro detekci hran. Když je zohledněna tato rovnice při konvoluci vstupního obrazu, výsledný obraz disponuje zvýrazněnými hranami i přes přítomnost šumu v originále.

Cannyho algoritmus se dělí na 5 kroků:

1. **Vyhlazování** Obrázky z kamer používaných ve fotbale robotů většinou nebývají nejvyšší kvality (a to ani nechceme, z důvodů náročnějšího zpracování velkých obrázků), proto obsahují značné množství šumu. Proto je použit Gaussův filtr, který obrázek zjemní, ale narozdíl od ostatních vyhlazovacích filtrů zachová viditelné hrany. Použitá odchylka (nebo také velikost jádra) Gaussova filtru záleží na potřebách a její velikost ovlivní sílu "rozostření". Zvětšováním odchylky je sice zmenšen dopad šumu, ale je zde možnost ztráty přesnosti a drobných detailů obrazu.

(a) Originální obrázek

(b) po aplikaci Gaussova filtru

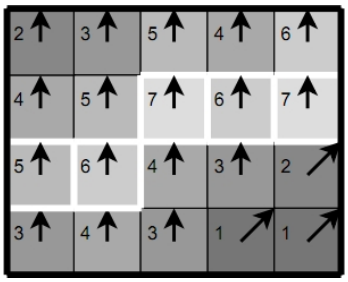
Obrázek 3: Cannyho hranový detektor: Gaussův filtr [9]

2. **Nalezení gradientů a jejich směr a velikost** Tedy klasická detekce hran. Pro tento krok je výhodné použít výše popsaný Sobelův detektor hran, pomocí kterého se dá vypočítat sílu a směr gradientů. Operátory detekce hran (např. Sobelův nebo Prewittův operátor) vracejí hodnotu první derivace v horizontálním (G_x) a vertikálním (G_y) směru. Pomocí těchto hodnot je zjištěn směr a síla gradientu:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{3.6}$$

$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right) \tag{3.7}$$

3. **Potlačení nemaximálních odezev** V tomto kroku se odstraňují body, jejichž síla gradientu není lokálním maximem. Toho se docílí porovnáním síly každého pixelu se silou jeho sousedů ve a proti směru gradientu. Nejdříve pro urychlení dojde k zaokrouhlení směru na nejbližší násobek $45\,^\circ$. Poté je provedeno porovnání se sousedními pixely (např. pokud je směr gradientu $0\,^\circ$, zohledněn je jeho levý a pravý soused). Není-li síla gradientu v daném pixelu větší než v sousedních, je potlačen (resp. síla je nastavena na 0 a pixel není součástí hrany). Jak je možno vidět na ilustraci č. 4 z [9] (šipkami uvedeny směry gradientů jednotlivých pixelů, čísla představují jejich sílu) pixely, které nejsou bíle orámovány byly poltačeny. V tomto kroku tedy dochází ke ztenčení hran a potlačení pixelů, které do hran nepatří.



Obrázek 4: Ilustrace poltačení nemaximálních odezev [9]

4. **Dvojité prahování** Předposlední krok má za cíl odstranit přebytečný šum, který byl v předchozích krocích detekován jako hrany. K tomu jsou použity dvě prahové hodnoty. Obrázek je procházen pixel po pixelu a pro každý je vykonáno následující: pokud je síla pixelu větší než vyšší práh, je považován za "silný". Je-li hodnota pixelu mezi oběma prahy, je označen jako "slabý". Pixely, které svou silou nedosahují na nižší práh, jsou potlačeny.

(a) Sobelův hranový detektor

(b) Potlačení nemaximálních odezev

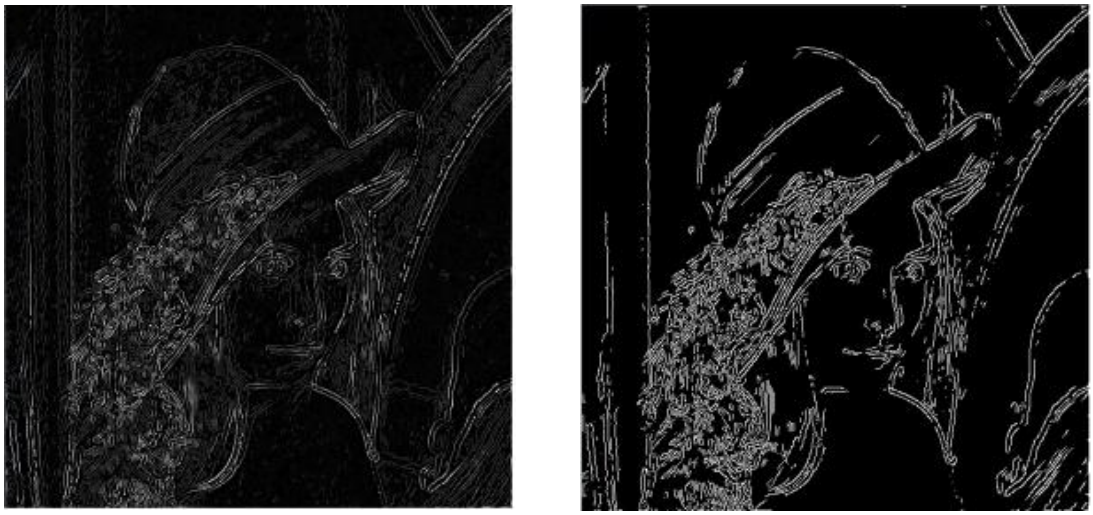
Obrázek 5: Cannyho hranový detektor: Sobel, Potlačení nemaximálních odezev [9]

5. **Hystereze** Po posledním kroku se v obrázku stále mohou vyskytovat body, které by neměly být součástí hrany. Hystereze spočívá v kontrole všech pixelů, které byly při dvojitém prahování označeny jako slabé. Ty jsou součástí hrany pouze tehdy, když přímo sousedí (okolních 8 bodů) se silným pixelem. Není-li tato podmínka splněna, pixel je potlačen a není tedy hranou.

3.2.3 Houghova transformace

Na segmentaci je možno se také dívat jako na hledání objektů známých tvarů a velikostí (samozřejmě pouze pokud jsou známé). Klasickým příkladem je hledání specifických objektů na leteckých snímcích nebo nalezení fotbalového míče na hřišti. Jeden z mnoha způsobů obnáší pohyb masky, jež má požadovanou velikost a tvar, nad obrazem a hledání velké podobnosti s objekty. Tento postup ovšem není příliš vhodný kvůli zkreslení tvaru, otočení nebo jiné velikosti objektů, která může nastat při digitalizaci scény. Jedna z efektivních metod jak řešit tento problém je právě Houghova transformace, jež může být použita i při detekci částečně překrývajících se objektů (míč je částečně schován za hráčovou nohou apod.)

Jedna z představ o této metodě je tato: máme za úkol najít na bílém obrázku černou kružnici o daném poloměru r. Pro nalezený černý pixel vytvoříme soubor možných středů původní kružnice. Tento soubor nám vytvoří novou kružnici o stejném poloměru jako má hledaná. Pokud to takto uděláme u všech černých pixelů, můžeme každému pixelu přiřa-



(a) Dvojité prahování (b) Konečný obrázek

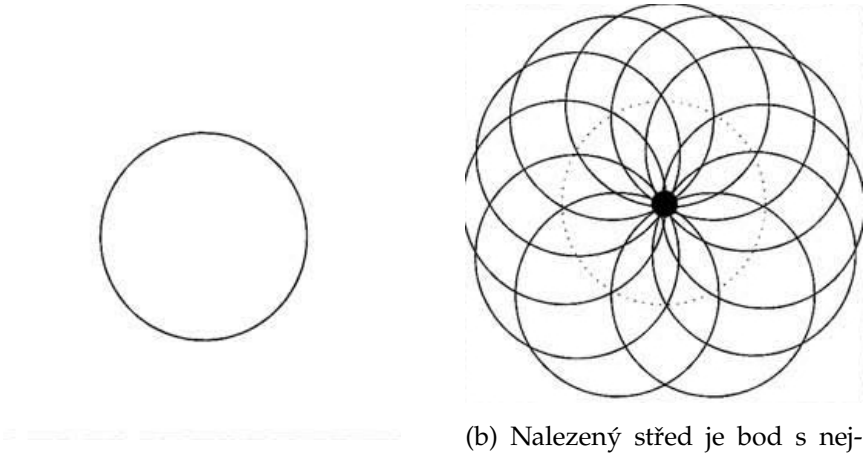
Obrázek 6: Cannyho hranový detektor: Dvojité prahování, Hystereze [9]

dit frekvenci (četnost) s jakou se vyskytuje v námi vytvořených kružnicích. Střed původní kružnice je poté bod s největší frekvencí výskytu ve "středových"kružnicích (jak je vidět na obrázku 7). Tím dostaneme střed, jež nám se známým poloměrem dokončí kompletní segmentaci. Tímto způsobem lze detekovat i objekty (v našem případě kružnice), které jsou částečně zakryty.

Základní myšlenku Houghovy transformace je možné vidět v problému detekce jednoduché přímky na obrázku. Přímka je definována dvěma body $A = (x_1, y_1)$ a $B = (x_2, y_2)$. Všechny přímky procházející bodem A jsou dány rovnicí $y_1 = kx_1 + q$ pro dané hodnoty k a q. Tuto rovnici lze vyjádřit v prostoru daném parametry (k, q), kde všechny přímky procházející bodem A mají rovnici $q = -x_1k + y_1$. Podobně přímky procházející bodem B jsou vyjádřeny rovnicí $q = -x_2k + y_2$. Jediný společný bod obou přímek v pomocném prostoru (k, q) je bod, který v původním prostoru x, y představuje jedinou přímku, procházející body A a B.

To znamená, že každá z přímek na obrázku je reprezentována jedním bodem v prostoru (k, q) a kterákoliv část přímky je transformována do toho samého bodu. Hlavní myšlenka detekce přímých čar v obraze je najít všechny takové pixely, které mohou náležet jedné z těchto čar. Poté je možno transformovat všechny čáry, které by mohly procházet těmito body do odpovídajících bodů v prostoru (k, q) a nalézt body v tomto prostoru, které nejčastěji vzešly z Houghovy transformace přímek $y = ax + b$ v obraze.

Použití rovnice přímky ve tvaru $y = kx + q$ je však vhodné pouze pro vysvětlení



(a) Původní obrázek kružnice

(b) Nalezený střed je bod s největší frekvencí výskytu v pomocných kružnicích

Obrázek 7: Houghova transformace: detekce kružnice [10]

Houghovy transformace a jejích principů. V praxi tento přístup působí problémy u detekce vertikálních přímek, kde směrnice k se blíží nekonečnu, proto Duda a Hart [11] navrhli využití normálového tvaru rovnice přímky:

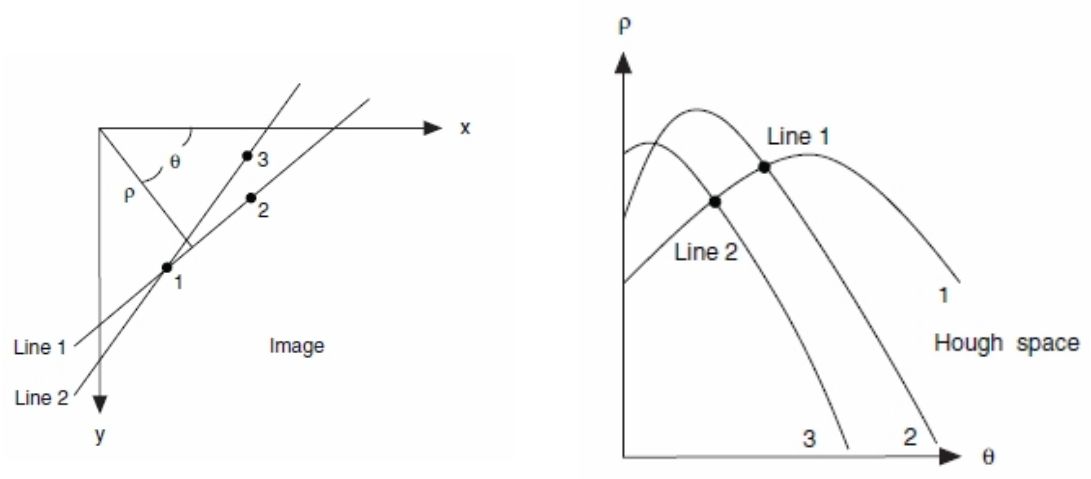
$$\rho = x \cos\theta + y \sin\theta \tag{3.8}$$

kde ρ vyjadřuje délku normály od přímky k počátku systému souřadnic a θ je úhel mezi normálou a osou x. Každý bod obrázku vytvoří (stejným způsobem jako v předchozím teoretickém příkladě) v parametrickém systému (ρ, θ) sinusoidu. Pokud by se jednalo o obraz s jedinou přímkou (resp. úsečkou), tyto sinusoidy by měly průnik v jediném bodě, který při dosazení do rovnice 3.8 dá rovnici přímky v normálovém tvaru a její detekce je tedy hotová. V klasickém obrázku může být mnoho úseček a proto se hledají všechny takové průsečiky, neboli lokální maxima protnutých sinusoid v grafu (viz. obrázek 8).

Houghova transformace je využívána některými týmy soutěžící v *Middle sized* lize fotbalu robotů, kde se stará o detekci hřiště z pohledu robota a jeho natočení a pozici [13]. Je možno ji použít také pro nalezení míče na hřišti pomocí detekce kružnic, kde se velmi hodí schopnost detekce částečně zakrytého míče [14].

3.3 Regionální segmentace (region-based)

Cíl předchozích dvou skupin bylo nalézt hranice mezi regiony. V této kapitole jsou popsány metody, které tvoří regiony obrazu přímo. Je jednoduché vytvořit regiony z jejich



(a) Přímky v prostoru (x, y)

(b) Transformace bodů do Houghova prostoru)

Obrázek 8: Houghova transformace: detekce přímky [12]

hranic a stejně tak je jedoduché najít hranice existujících regionů. Obrazy po segmentaci pomocí těchto dvou metod ovšem nebývají úplně stejné, proto je někdy nejlepší kombinace obou. Postupy založené na tvorbě regionů (region-growing) jsou obecně lepší u obrázků s velkým procentem šumu, kde hrany bývají těžko detekovatelné. Jsou založené na homogenitě pixelů a je to hlavní kritérium pro segmentaci u tvorby regionů. To znamená že jejich pixely mají podobné vlastnosti, např. barevný odstín, tvar, odstín šedi apod. Vybraná vlastnost ovlivní složitost a množství informací, jaké budeme mít k dispozici při dané implementaci metody.

K obecné definici regionů uvedené v kapitole 3.1 je nutné přidat další podmínky, se kterými bychom měli počítat při jejich tvorbě:

$$H(R_i) = \text{PRAVDA}, \quad i = 1, 2, \ldots, S, \tag{3.9}$$

$$H(R_i \cup R_j) = \text{NEPRAVDA}, \quad i \neq j, \quad R_i \text{ přímo sousedí s } R_j, \tag{3.10}$$

kde S je celkový počet regionů v obrázku a $H(R_i)$ je binární vyjádření homogenity regionu R_i. Výsledné regiony obrazu musí být homogenní a maximální, což znamená, že kritérium homogenity by neplatilo při spojení se sousedním regionem.

3.3.1 Spojování regionů (Region merging)

Metoda, která každého napadne jako první, je začit s tvorbou regionů takto. Na začátku považujeme každý pixel jako samostatný region. Takovéto regiony určitě nesplňují druhou podmínku 3.10, a proto můžeme regiony spojovat, dokud podmínka 3.9 bude splněna. Algoritmus skončí jakmile nenajde další regiony, které by vyhovovaly podmínce homogenity a mohly by být tedy spojeny.

Tento postup představuje základ pro segmentaci spojováním regionů. Konkrétní metody se liší v pojetí počáteční segmentace a v kritériích pro spojení. Výsledek této segmentace je většinou také závislý na tom, kde se spojováním začneme. Což znamená, že když budeme regiony spojovat například od levého horního rohu nebo pravého spodního rohu, budou se výsledky těchto dvou postupů lišit. To se děje, protože pořadí spojení může způsobit, že se podobné regiony R_1 a R_2 nespojí, jelikož dřívější spojení použilo region R_1 a jeho nové vlastnosti mu nedovolí se spojit s R_2. Pokud by se začínalo se spojením na druhé straně, je možné, že toto spojení by proběhlo.

Nejjednodušší metody začínají segmentaci za použití regionů o velikost 2×2, 4×4 nebo 8×8 pixelů. V případě černobílého obrazu jsou regiony popsány například pomocí regionálního histogramu. Vyvozené vlastnosti z těchto histogramů jsou poté porovnány a pokud jsou podobné, je z těchto dvou regionů vytvořen nový. Jinak jsou označeny jako nepasující. Spojování pokračuje mezi všemi sousedy včetně nově vytvořených. Pokud region už nemůže být spojen s žádným dalším, je označen jako konečný. Spojování pokračuje do doby, než jsou všechny regiony takto označeny.

Důležitou částí těchto algoritmů je způsob jak určit počáteční body (semínka, angl. seed), od kterých se bude tvorba regionů rozvíjet. Jak je popsáno v [15] , může v některých případech pomoci speciální hardware; zařízení, které provede prahování obrázku do několika barev a je možné využít bodů jednotlivých barev jako semínek. Pokud takovýto hardware není k dispozici, je potřeba využít důležitých vlastností, které by mělo takové semínko mít, aby byla segmentace smysluplná. Například v [16] je v jejich metodě použito následujících 3 vlastností k automatické volbě semínek:

- Pixel představující semínko musí mít velkou míru podobnosti s jeho sousedy
- Pro každý předpokládaný region musí být vytvořeno alespoň jedno semínko
- Semínka různých regionů nesmí být sousedé

Míra podobnosti se sousedními pixely je uvedena jako:

$$\sigma_x = \sqrt{\frac{1}{9}\sum_{i=1}^{9}(x_i - \overline{x})^2} \tag{3.11}$$

kde x může být hodnota Hue, Saturation nebo Intensity (barevný model HSI) a střední hodnota $\overline{x} = \sum_{i=1}^{9} x_i$. Celková směrodatná odchylka je $\sigma = \sigma_H + \sigma_S + \sigma_I$.

3.3.2 Ostatní metody založené na regionech

Kromě spojování regionů existuje několik dalších typů postupů, které se ve fotbale robotů příliš nepoužívají, ale je vhodné o nich něco vědět.

- **Rozdělování regionů (Region splitting)** je opak spojování regionů a začíná s jediným regionem, který zabírá celý obrázek. Tento velký region je poté rozdělován tak, aby splňovaly všechny 3 výše uvedené podmínky. Tento proces je opačným postupem ke spojování, ale jeho výsledek je mnohdy velmi rozdílný, ikdyž je použito stejné kritérium homogenity. Například když bychom chtěli rozdělit klasickou šachovnici a jako kritérium bychom měli průměrné rozdíly v odstínech šedé. Při segmentaci rozdělováním regionů bychom skončili hned při prvním kroku, protože celý obrázek je vůči našemu kritériu homogenní, stejně jako jeho podregiony (bereme-li klasické rozdělení na stejné čtvrtiny apod.). Spojováním regionů bychom ovšem došli k rozdělení na jednotlivá pole šachovnice a segmentace by byla úplná.

- **Spojování a rozdělování regionů** je kombinací obou výše zmíněných postupů s výhodami spojování i rozdělování regionů. Tato metoda využívá pyramidového pojetí, kde pokud je nějaký z regionů nehomogenní, rozpadne se na 4 stejně velké subregiony. Naopak když se v obrázku vyskytnou 4 sousední regiony, které vykazují homogenitu, jsou spojeny v jeden. Úvodní rozdělení regionů je určeno buď předem ručně, nebo automaticky. A protože používáme jak rozdělování, tak spojování regionů, nemusí startovní regiony splňovat žádnou ze dvou podmínek 3.9 a 3.10

- **Segmentace rozvodím (Watershed segmentation)** je velmi známou a často využívanou metodou v topografických aplikacích. Obrázek si představíme jako krajinu, kde intenzity odstínu šedé (popř. některé z barevných složek) představují nadmořskou výšku. Lokální minima intenzity jsou poté jakési základny *vodní nádrže*. Existují dva hlavní přístupy k segmentaci zaplavováním. První hledá cestu *z kopce* dolů

(intenzita šedé se musí zmenšovat) směrem k lokálnímu minimu. Region je poté definován jako sada pixelů se stejným konečným lokálním minimem. Druhý přístup začíná plnit *nádrž* (region) od lokálních minim. Jakmile by se měly dva regiony do sebe prolnout, čára směrem k nejvyššímu lokálnímu maximu je jejich hranicí. První typ algoritmu je výpočetně relativně náročný a proto se využívá převážně druhý přístup. Příklad segmentace pomocí druhého postupu uvádí [17].

3.4 Ztenčování (thinning)

Jednou z důležitých a velmi používaných metod pro segmentaci obrazu je ztenčování hran. Tyto algoritmy většinou následují po hranové detekci a jejich cílem je zredukovat objekty na jejich kostry. Ač jde o často zmiňovaný pojem, stále neexistuje sjednocená definice kostry objektu. Jedná se o relativně malý soubor pixelů, kde žádný z nich není zanedbatelný, protože dohromady představují základní tvar objektu. Kostra zachovává všechny důležité vlastnosti původního objektu, což u našeho tématu fotbalu robotů jsou pozice, velikost a předpokládaný směr "pohledu".

Většina moderních ztenčovacích algoritmů vychází z dlouho známého principu a v poslední době se výzkumy zabývají spíše rychlostí než přesností algoritmu. V této kapitole si představíme základní postup a poté jedno z možných vylepšení pro urychlení procesu. Pro začátek je však nutné uvést několik důležitých faktů, které je potřeba mít na paměti:

- Ne všechny objekty mohou nebo by měly být ztenčeny. Postup ztenčení je vhodný pro objekty složené z čar. Naopak nehodí se pro objekty, které zabírají velkou plochu nebo jsou "plné". Například hrany robota můžeme ztenčit, ale kdyby byl robot představován dvěmi vyplněnými čtverci, neexistuje vhodný postup, jak takovýto objekt ztenčit.
- Žádná ze ztenčovacích metod není univerzální. Někde se hodí přesnost, jinde rychlost.
- Ztenčení znamená získání kostry objektu. Ta by měla vycházet z objektu a neměla by být definována postupem algoritmu. Ztenčení by nemělo znamenat rekurzivně mazat vnější vrstvy pixelů.

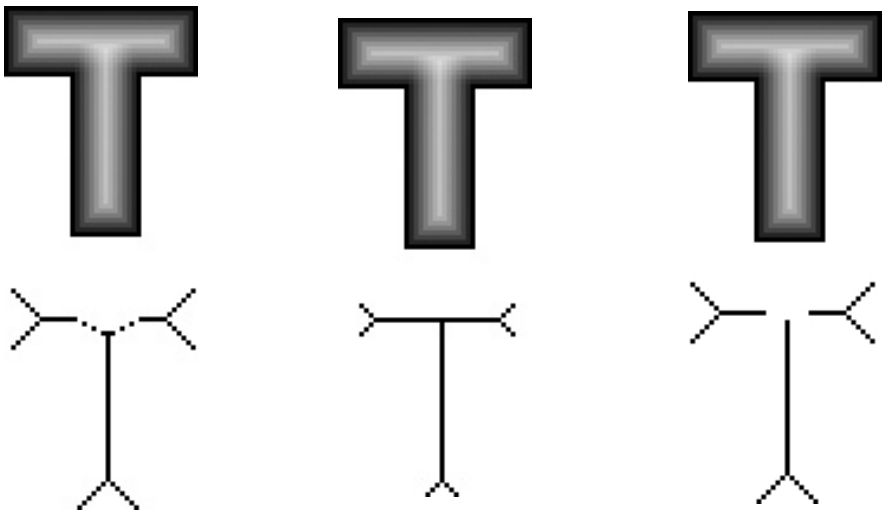
3.4.1 Transformace středové osy

První zmínky o definici kostry jsou od H. Bluma [18] , který zavedl pojem *Funkce mediální osy* (MAF - Medial Axis Function). Původní myšlenka byla tato: MAF považuje všechny

hraniční pixely za zdroje vln. Když se dvě vlny setkají, vyruší se a vytvoří *roh*. Soubor všech takových rohů nám dá kostru.

Jedna z možností jak najít středovou osu využívá hrany objektu. Pro každý bod objektu (A) najdeme nejbližší bod hrany (B). Pokud nalezneme více takových bodů (B), náleží (A) středové ose. Všechny takové body nám dají samotnou osu. K tomuto postupu je ovšem potřeba velkého rozlišení obrázku, jinak euklidovské vzdálenosti nebudou stejné, ikdyž by měly být, což zapříčiní nesprávnou identifikaci bodů kostry.

V praxi se dá přibližná středová osa jednodušeji určit ve dvou krocích. Nejprve je vypočítána vzdálenost každého pixelu objektu od nejbližšího hraničního bodu. Poté se použije Laplaceův operátor na vzdálenosti a pixely s velkými hodnotami by měly být součástí osy. Velmi také záleží na tom, jak je počítána potřebná vzdálenost pixelu od bodu hrany (viz. obrázek č.9). Jsou 3 způsoby: euklidovský a dva, které souvisí s nejkratší cestou od bodu k bodu. První je 4-směrová, kde se bere nejkratší možná cesta z A do B za použití pouze vertikálního a horizontálního posunu. Druhá 8-směrová je totéž s pomocí 8 základních směrů.



(a) 4-směrová v. (b) 8-směrová v. (c) Euklidovská v.

Obrázek 9: Ukázka efektu různého počítání vzdálenosti na středovou osu [6]

Tyto kostry nemají sice přesný tvar T, ale to ani nemusí. Důležité je že zachovávají základní rysy původního obrazu. Problém u tohoto postupu je, že stačí odebrat jeden pixel z obrazu a kostra se může radikálně změnit. Ztenčení pomocí transformace středové osy nemá kvůli své pomalé rychlosti valné praktické využití. Jak uvádí [6], z teoretického hlediska tato metoda dává základy většině ostatních algoritmů, a proto je důležité ji znát.

3.4.2 Zhang-Suen algoritmus

Jeden z nejpoužívanějších ztenčovacích algoritmů, navržen Zhangem [19] v roce 1984. Jedná se o rychlý a jednoduše implementovatelný algoritmus, který je roky využíván jako základ pro porovnávání rychlostí a efektivit ztenčovacích metod. Patří do skupiny iterativních algoritmů, které postupně mažou vrstvy pixelů objektu, dokud nezůstane pouze kostra. Průběh algoritmu je paralelní. S dostatečným počtem procesorů, popřípadě při použití technologií jako je CUDA [20] (využití výpočetních jader grafické karty pro paralelizaci), je možné provést ztenčení velmi rychle.

Proces je rozdělen do dvou částí, kde v první je pixel *P(i, j)* smazán, pokud splňuje následující 4 podmínky:

1. Jeho počet připojení je 1. To znamená, kolik objektu může bod spojovat. Jestliže se rovná 1, po smazání se nerozpojí žádná hrana.
2. Má nejméně 2 a nejvíce 6 černých sousedních pixelů.
3. Aspoň jeden z trojice pixelů *P(i-1, j)*, *P(i, j+1)* a *P(i+1, j)* je bílý.
4. Aspoň jeden z trojice pixelů *P(i, j+1)*, *P(i+1, j)* a *P(i, j-1)* je bílý.

Druhá část je stejná až na krok 3 a 4:

3. Aspoň jeden z trojice pixelů *P(i-1, j)*, *P(i, j+1)* a *P(i, j-1)* je bílý.
4. Aspoň jeden z trojice pixelů *P(i-1, j)*, *P(i+1, j)* a *P(i, j-1)* je bílý.

a opět jsou všechny vyhovující pixely smazány. Pokud na konci jedné z částí nebyly smazány žádné pixely, algoritmus končí a kostra je hotová.

Výsledek a efektivita Zhang-Suen algoritmu je vidtelná na obrázku č.10, kde byl na objekt robota použit právě tento, lehce upravený algoritmus.

(a) (b)

Obrázek 10: Ukázka Zhang-Suen algoritmu před a po ztenčení

4 Návrh a implementace vlastního postupu

Jak bylo již uvedeno v úvodu předchozí kapitoly 3.1, jedním z důležitých rozhodnutí, než začneme cokoliv zpracovávat, je výběr barevného prostoru ve kterém budeme pracovat. Tato volba závisí na kvalitě kamery, která snímá herní plochu, kvalitě a rovnoměrnosti osvětlení a dalších potřebách dané aplikace.

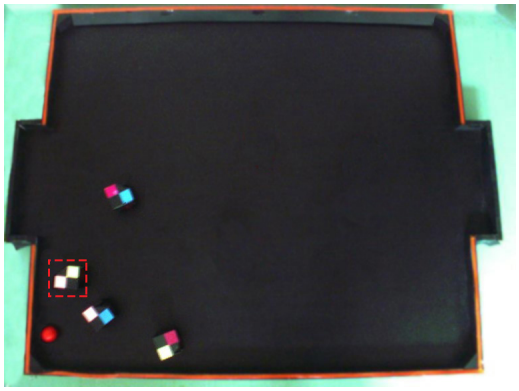
Nejznámějším systémem je RGB (Red, Green, Blue), který míchá barvy jako kombinaci červené, zelené a modré. Tento přistup má ovšem několik nedostatků pro použití v oblasti vize robotů a jejího digitálního zpracování. Právě ve fotbale robotů jsou důležité objekty na hřišti označeny specifickými barvami (např. míček je červený, jeden z týmů žlutý apod.). Proto potřebujeme aby náš barevný systém byl dostatečně flexibilní a jednotlivé barvy byly od sebe jednoduše rozlišitelné. Nekonzistentní osvětlení objektů může některé barvy zesvětlit a to se v RGB systému nedá velmi dobře rozlišit.

V tomto projektu jsme proto zvolili barevný prostor HSL (Hue, Saturation, Lightness), který barvu vyjadřuje jako kombinaci odstínu, sytosti a světlosti. To nám umožňuje starat se primárně o odstín barvy a nemusíme zjišťovat a počítat, jaké kombinace červené, zelené a modré odpovídají světlejšímu odstínu hledané barvy. Díky tomu nám stačí pouze několik prahů, ve kterých počítáme s odchylkami, pro určení zda se například jedná o míč nebo hráče.

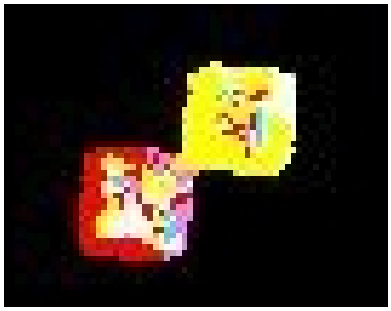
4.1 Detekce hran

Nejpoužívanější metodou segmentace je bezpochyby prahování 3.1, které převede obrázek do několika snadno rozlišitelných barev. Je to velmi rychlá metoda a v mnoha případech se jí vyplatí zařadit do procesu zpracování obrazu. Po několika testech jsem se rozhodl tento krok vynechat. Testovací obrázky (viz. obrázek 11a) ukazují na několik nevýhod prahování. Kvůli nedokonalému osvětlení místnosti a světlým barvám na identifikačních čtverečcích hráčů se stávají regiony nekonzistentní v barevném odstínu (viz. obrázek 11). Na tomto výřezu můžeme vidět že barvy se při digitalizaci z velké části nepodařilo zachovat, proto tento postup nemůžeme považovat za spolehlivý.

Jako první bod celého procesu analýzy jsem zvolil detekci hran. Konkrétně se jedná o Sobelův hranový detektor (více v kapitole 3.2.1). Z hranových detektorů je nejrychlejší a pro naše účely jeho nedostatky nezpůsobují problémy. Například Cannyho hranový detektor (více v kapitole 3.2.2 rozpozná hrany kvalitněji a přesněji, je ovšem oproti sobelu pomalejší a tudíž nepříliš použitelný. Rychlost Cannyho hranového detektoru je menší už jen proto, že Canny ve svém algoritmu využívá právě Sobelův detektor jako jeden z kroků. Ve svém postupu jsem použil paralelizovaný Sobelův detektor aplikovaný na



(a) Originální obrázek



(b) Výřez z obrázku po prahování

Obrázek 11: Ukázka nedostatků prahování na testovacím obrázku

barevný obrázek. Tento krok také provádíme v barevném prostoru RGB, protože nemusíme pracovat s přesnými barvami, ale jen s hodnotami jednotlivých složek. Algoritmus je následující:

1. Na pixel *P(i, j)* aplikujeme rovnice č.3.2 a 3.3, kde za I nahradíme barevnou hodnotu postupně každé ze složek systému RGB. Získáme tak 6 nových proměnných (R_x, R_y, G_x, G_y, B_x a B_y)

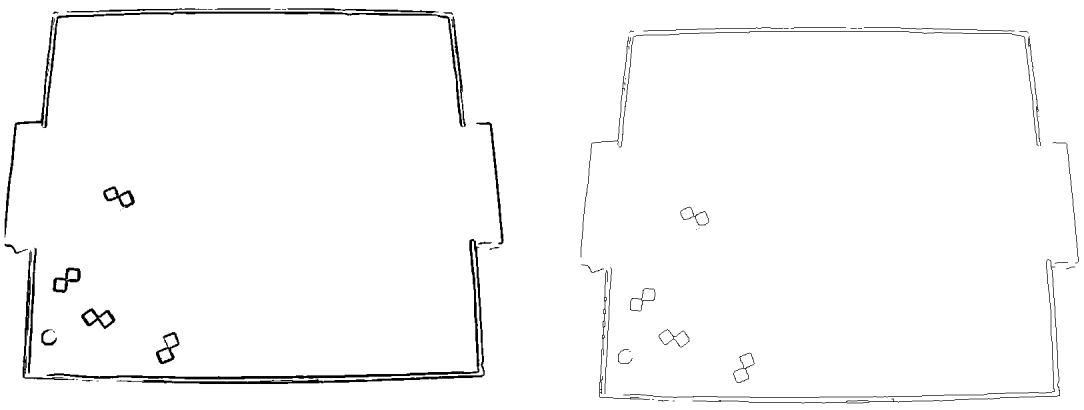
2. Porovnáme součty druhých mocnin (4.1) hodnot pro jednotlivé barevné složky s předem určeným prahem T pro zjištění zda se jedná o hranu.

$$R_x^2 + R_y^2 > T \quad \vee \quad G_x^2 + G_y^2 > T \quad \vee \quad B_x^2 + B_y^2 > T \tag{4.1}$$

3. Platí-li podmínka (4.1) jedná se o pixel náležící hraně.

4. Kroky 1.-3. opakujeme pro každý pixel obrázku

Je-li pixel hranový nebo ne, se ukládá do dvojrozměrného binárního pole. Hranové pixely představují 1 (resp. *PRAVDA*), ostatní 0 (*NEPRAVDA*). Toto pole poté využijeme v některých dalších krocích pro rychlejší přístup a zpracování informací, abychom nemuseli neustále pracovat s daty obrázku. Pro zrychlení jsem použil paralelní *for* cyklus na procházení řádků pixelů obrázku. Vždy se vyhodnocuje několik řádků najednou. To si můžu dovolit z důvodu, že u Sobelova detektoru neovlivní již vyhodnocené pixely výpočet ostatních. Výsledky Sobelova hranového detektoru jsou na obrázcích č. 2 a 12a.



(a) Testovací obrázek po aplikaci Sobelova detektoru hran

(b) Ztenčené hrany pomocí Zhang-Suena

Obrázek 12: Implementace: Sobelův detektor hran a Zhangovo-Suenovo ztenčování

4.2 Ztenčení detekovaných hran

Jak je patrné z obrázku po aplikaci Sobelova operátoru, detekované hrany stále nesplňují podmínky, které stanovil Canny (viz. kapitola 3.2.2). Hlavně třetí podmínku, která říká, že hrana nesmí být zdvojená (a vícevrstvá), tedy musí sestávat pouze z jednoho pixelu tam, kde je to možné. Zvolil jsem tedy cestu ztenčování hran. Průběh algoritmu je v podstatě stejný, který je uveden v kapitole 3.4.2. Výsledek po ztenčení hran je zobrazen na obrázku č.12b.

4.3 Lokalizace hřiště a branek

První krok pro úspěšné nalezení objektů (hráčů a míče) na hřišti, je vymezení hřiště samotného. Následující postup využívá faktu, že šířka mezi 2 černými pixely ve ztenčeném obrázku bude na každém řádku vždy největší uvnitř hřiště. Ve Small lize fotbalu robotů používají dvě kamery, kdy každá z nich snímá jednu polovinu hřiště. To naše je menší a proto se v našem sestavení nachází pouze jedna kamera. Hřiště tak zabírá valnou část celého obrázku a kamera je nastavena tak, aby v ideálním případě snímala pouze hřiště a to z pozice kolmo nad jeho středem. Cyklus pro nalezení největší šířky probíhá pro každý řádek pixelů obrázku následovně:

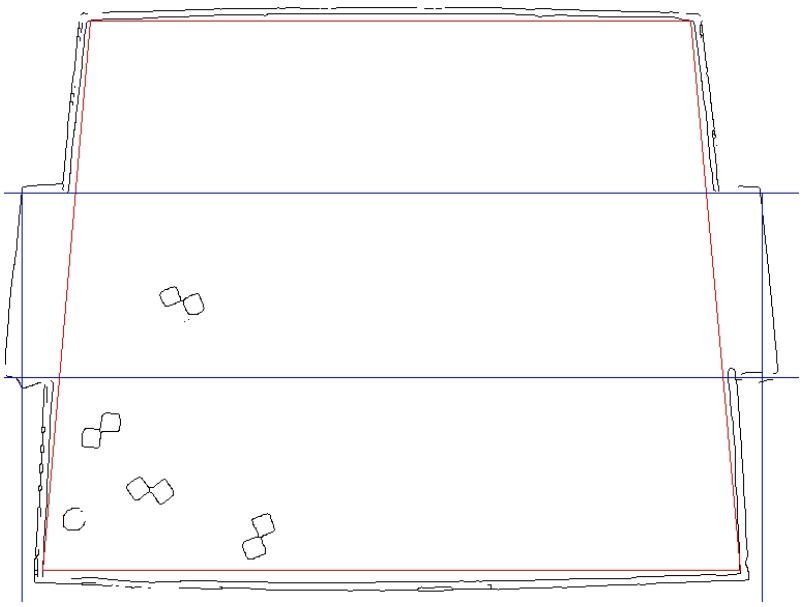
1. Procházíme pixely řádku dokud nenarazíme na jeden s černou barvou (v použitém binárním poli buňka s hodnotou *true*) a zapamatujeme si jeho horizontální pozici.

2. Pokračujeme do dalšího černého pixelu a vzdálenost mezi nimi je hledaná šířka.

3. Uložíme si největší nalezenou šířku na řádku.

Nyní máme seznam největších vzdáleností mezi 2 černými pixely. Najdeme první hodnotu větší než 0, abychom našli vrchní okraj hřiště. To má malý okraj u kterého předpokládáme, že je široký přibližně 10 pixelů. Přidáme tedy k indexu horního okraje dvojnásobek předpokládané šířky okraje, tedy 20. Od tohoto indexu poté postupně procházíme dříve nalezené vzdálenosti směrem zpět (tzn. k horní hranici obrázku). Pokud je rozdíl mezi dvěmi šířkami větší než 10, algoritmus končí a nalezli jsme horní okraj hřiště. Podobně to uděláme i pro spodní hranici. Díky zapamatovaným levým rohům a šířkám získáme všechny 4 rohy hřiště. Výsledek tohoto postupu můžeme vidět označen červeným čtyřúhelníkem na obrázku č.13

Pro nalezení hranic branek použijeme pomocný obrázek s vyznačeným hřištěm. Podobně jako u algoritmu pro lokalizaci hřiště využijeme vzdálenost mezi dvěma body. Opět začínáme z levé strany obrázku a hledáme vzdálenosti mezi černým a červeným pixelem, který značí hřiště. Ty které jsou větší než daný práh, jsou součástí branky. Tentokrát nám stačí pouze první a poslední takový řádek pro určení souřadnice y a jeden z levých (resp. pravých) pixelů pro souřadnici x. Nalezené branky můžeme vidět na obrázku č.13, kde jsou označeny modrými přímkami. Pro branky předpokládáme, že jsou obě ve stejné rovnině, tudíž nám na vertikální ose stačí zjistit hranice jedné z branek.



Obrázek 13: Implementace: Lokalizace hřiště a branek

4.4 Objekty na hřišti

Poslední část procesu zpracování obrazu je samotná analýza, neboli popis dění na hřišti. Napříč všemi týmy z lig fotbalu robotů se jedná o oblast, která je nejrozmanitější co se týká různorodosti použitých algoritmů. Nabízí se zde využití Houghovy transformace pro detekci čar a kružnic (3.2.3), ovšem v případě našeho rozestavení a stylu identifikace hráčů se jedná o zbytečně silný nástroj. Pro naše množství objektů na hřišti je to výpočetně náročná operace, takže jsem volil raději algoritmus vytvořený na míru použitým prostředkům.

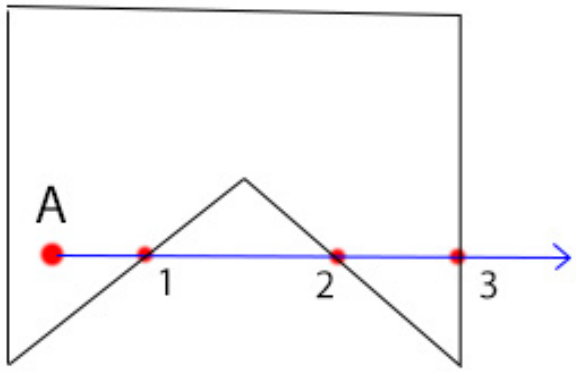
4.4.1 Bod uvnitř hřiště

Prvním předpokladem pro to, aby objekt byl na hřišti je, že všechny jeho pixely jsou uvnitř hřiště nebo branky. Zjištění zda-li je bod uvnitř mnohoúhelníku je klasický problém a existuje pro něj několik algoritmů. Jedním z rychlých a jednoduchých postupů je algoritmus *vrhání paprsků*, neboli ray casting.

Předpokládejme mnohoúhelník jako na obrázku č.14. Když vyneseme přímku (paprsek) z bodu A libovolným směrem, počet průsečíků s hranami mnohoúhelníku nám určí, zda v něm bod leží. Můžeme si to představit jako binární přepínač, který se aktivuje při každém protnutí hrany s přímkou, a který začíná ve stavu *false* (nepravda). Vememe-li si bod A, tak při prvním protnutí se paprsek dostane ven z objektu, druhé protnutí paprsek dostane zpět dovnitř a třetí znovu ven. Kdybychom posunuli bod A po přímce doprava mezi první dva průsečíky ven z objektu, zbyly by pouze 2 a bod bychom označili jako mimo mnohoúhelník. Pokud je tedy počet průsečíků lichý je bod uvnitř. Při sudém počtu bod není součástí objektu.

Pro tento účel jsem si vypůjčil algoritmus, který uvedl W. Randolph Franklin v článku [21]. Tento postup je aplikovaný výše zmíněný *ray-casting*, zredukovaný na potřebné minimum výpočtů a porovnání. Cyklus porovná bod postupně se všemi hranami hřiště a pokud je vyhověno podmínce v [21], přepne se přepínač, který udává, je-li bod uvnitř hřiště.

Ještě před vykonáním tohoto algoritmu je zkontrolována jedna podmínka, která celý proces výrazně urychlí. Tvar hřiště je v ideálním případě obdélník, při mírném náklonu kamery se z něj stane lichoběžník. Můžeme si představit obdélník jako na obrázku č.15. Zjištění, zda je bod uvnitř tohoto obdélníku obnáší pouhé porovnání souřadnic bodu se souřadnicemi 4 vrcholů. Stejně tak můžeme zjistit, je-li bod v brance; jednoduchým porovnáním souřadnic s brankou.



Obrázek 14: Implementace: Bod leží uvnitř mnohoúhelníku

Obrázek 15: Implementace: Zjednodušení hledání bodu v lichoběžníku

4.4.2 Lokalizace relevantních objektů

Nalezení hráčů a míče spočívá v rozdělení pixelů na jednotlivé objekty. Toho jsem docílil pomocí rekurzivního algorimu, který postupně prochází sousední připojené pixely. Průběh algoritmu je následující (stále pracujeme s binárním polem, kde černá = *PRAVDA* a bílá = *NEPRAVDA*):

1. Procházíme všechny pixely v rozmezí horního a spodního okraje hřiště. Je-li bod černý a součástí hřiště pokračujeme krokem 2, pokud neplatí jedna z podmínek přejdeme na další pixel.

2. Uložíme si nalezený pixel do seznamu pro nový objekt a smažeme nalezený pixel (označíme ho jako bílý).

3. Nyní postupně procházíme všech 8 sousedních pixelů. Pokud narazíme na černý pixel, který je součástí hřiště, vrátíme se na krok 2.

Tímto vytvoříme rekurzi, která při nalezení uloží a smaže daný pixel a díky tomu se nevytvoří nekonečná smyčka. Získaný seznam objektů poté profiltrujeme tak, že necháme pouze objekty s větším počtem bodů. To je kvůli možnému šumu, který se může objevit jako důsledek detekce hran a jejich ztenčení.

4.5 Určení detailních informací o objektech

Když máme seznam nalezených objektů, které se nacházejí na hřišti, musíme je rozlišit na hráče a míč. Poté potřebujeme zjistit kde přesně se každý objekt nachází, kam každý hráč směřuje, do jakého týmu patří a jakou v něm zastává roli (obránce, útočník apod.).

4.5.1 Rozlišení hráčů a míče

Hráči se v našem sestavení vyznačují dvojicí barevných obdélníků (viz. obrázek č.10, 11). Tento tvar nám zaručuje, že bude mít více pixelů, než necelá kružnice, která označuje míč. Pro míč tedy platí, že má nejméně bodů ze seznamu relevantních objektů. Zbytek jsou hráči.

4.5.2 Pozice, směr a velikost objektů

Pozice hráčů a míče je určena jejich středem a rozměry. Z bodů každého objektu získáme průměr hodnot pro obě souřadnice x a y. Tyto dvě hodnoty nám dají střed každého z objektů. Pro velikost nám stačí najít minima a maxima zvlášť pro souřadnice x a y. To stejné platí pro hráče i pro míč.

Pro určení směru natočení hráčů využijeme vlastností dvojobdélníku, který je představuje. Rozdělíme si hráče na 4 kvadranty podle jeho středu jako na obrázku č.16a. Zjistíme v každém kvadrantu počet pixelů, které mají vzdálenost od od větší než 1. Nyní porovnáme počet pixelů v prvním a nultém kvadrantu. Je-li počet v prvním kvadrantu větší, objekt směřuje do červeného výřezu na obrázku č.16b. Naopak když je větší počet pixelů v nultém kvadrantu, hráč směřuje do výřezu zelené barvy. Během tohoto procesu si také zaznamenáváme hodnotu Hue (odstínu barvy) každého bodu pro každý kvadrant zvlášť. To nám pomůže později.

Přibližný úhel směru hráče určíme tímto jednoduchým postupem (na ukázkovém obrázku má první kvadrant vyšší počet pixelů, proto se pohybujeme v červeném výřezu):

1. Dvojobdélník má vždy 8 stran. Díky tomuto poznatku můžeme vypočítat průměrnou velikost jedné strany. Z této průměrné hodnoty poté vypočteme $\frac{4}{5}$, což je ve výsledku $\frac{p}{10}$, kde p je počet bodů objektu. Výslednou hodnotu si označíme r. Tato úprava je kvůli nepřesnostem, kdy strana může mít podprůměrnou velikost a úhel je poté příliš nepřesný. Hodnota r nám udává pořadí bodu na úsečce, která určí možný směr hráče.

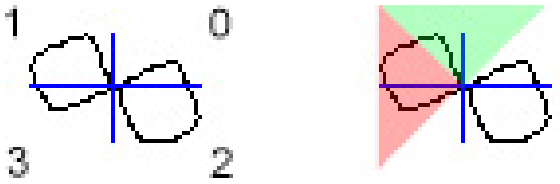
2. Začneme od středu procházet body, které patří do výřezu. Když se počet nalezených pixelů rovná hodnotě r, cyklus končí.

3. Ze souřadnic posledního nalezeného bodu v kroku 3 a středu objektu můžeme vypočítat jeden ze dvou možných úhlů pro směr hráče pomocí rovnice 4.2:

$$\arctan 2(s_x - b_x, s_y - b_y), \tag{4.2}$$

   kde s_x a s_y jsou souřadnice středu hráče a b_x a b_y jsou souřadnice bodu nalezeného v kroku 3. Funkce arctan2(y, x) je hodnota shodná s úhlem sevřeným mezi osou x a průvodičem bodu (x, y).

Při počítání bodů kvadrantů jsme si zaznamenávali hodnotu odstínu barvy každého pixelu. Ze seznamu těchto hodnot pro každý kvadrant určíme medián, protože není ovlivněn extrémními hodnotami, které v obdélníku nastanou jak je vidět na obrázku č.11b. Tyto mediány poté použijeme pro přesnější určení úihlu směřování hráče. Je-li odstín prvního (resp. nultého) kvadrantu v rozsahu týmových barev, musíme otočit úhel směřování o $180°$, jelikož předpokládáme, že označení týmu má hráč vždy vzadu. Z tohoto poznatku můžeme poté snadno rozdělit hráče do týmů a jejich pozic, kdy předem dané barevné odstíny rozpráhujeme s dostatečnými ochylkami. Barvy musíme zadat do programu předem a měly by být v rámci hráčů jednoznačně rozlišitelné (minimálně $50°$ rozdíl na stupnici barevných odstínů - *Hue*).



(a) Rozdělení hráče na kvadranty

(b) Výřezy možných směrů hráče

Obrázek 16: Implementace: Ilustrační výřezy pro určení úhlu směřování hráče

Sobelova detekce hran	13ms
Zhang-Suen ztenčení	24ms
Lokalizace hřiště a branek	3ms
Lokalizace relevantních objektů na hřišti	5ms
Detekce detailů o hráčích	<1ms
Celkem	<45ms

Tabulka 1: Přibližné rychlosti zpracování jednotlivých úseků

Výsledná knihovna obsahuje hlavní funkci, která na vstupu vyžaduje obrázek, který chceme analyzovat. Snímky dostáváme z kamery umístěné nad hřištěm. Také je třeba poskytnout informace o barvách hráčů, podle kterých je budeme rozdělovat do týmů a pozic v nich. Výstup zahrnuje informace o pozici každého hráče a míče, který se vyskytuje na hřišti nebo v brance. Součástí těchto informací je také přibližné natočení hráčů (s odchylkou zhruba $5°$), což je uvedeno jak ve stupních tak v radiánech vzhledem k horizontální ose. Knihovna také poskytuje funkci, která kontroluje zda míč není v brance, což by znamenalo, že jeden z týmů skóroval. Na obrázku č.17a je jeden z testovacích snímků použitého v této práci. Výsledné souřadnice jednotlivých robotů a míče jsou ukázány v tabulce 17b.

Rychlost vykonávání analýzy jednotlivých snímků se pohybuje na hranici zpracování v reálném čase. Pro naše potřeby je dosažená rychlost přes 20 snímků za sekundu dostačující (viz. tabulka 1).

(a) Testovací obrázek

Objekt	Střed	Rozsah X	Rozsah Y	Natočení	Tým	Pozice
Hráč 1	(383, 194)	366 - 400	178 - 210	$-94,4°$	1	2
Hráč 2	(548, 219)	529 - 567	204 - 234	$-14°$	2	1
Hráč 3	(540, 293)	525 - 556	274 - 313	$15°$	2	2
Hráč 4	(344, 373)	321 - 367	359 - 388	$118°$	1	1
Míč	(387, 231)	377 - 398	221 - 241			

(b) Tabulka s výstupními hodnotami

Obrázek 17: Příklad vstupního obrázku a výstupu analýzy

5 Závěr

Cílem mé bakalářské práce bylo prostudovat aktuálně používané metody analýzy obrazu fotbalu robotů a navrhnout novou nebo optimalizovat stávající metodu. Měl být vytvořen modul pro zpracování obrazu, ze kterého byly získány informace o pozici míče, hráčů a jejich týmové příslušnosti (resp. roli v týmu).

Úvodní kapitola se věnovala historii a struktuře fotbalu robotů a analýze jako takové. Poté jsme si představili několik základních metod pro zpracování obrazu. Nakonec byl popsán nově sestavený postup, který kombinoval výhody všeobecně používaných metod. Během tohoto návhru proběhlo také srovnání s těmito základními postupy.

Pro samotné nalezení objektů na hřišti a určení jejich pozice a směru jsem vytvořil vlastní algoritmus na míru našemu systému značení. Jedná se o proces, který rámci několika milisekund nalezne hráče a míč díky jejich společným vlastnostem. Nebylo tedy třeba využívat složitých matematických výpočtů, které by ve velkém množství pixelů (až 500 tisíc) mohly zahltit průběh a výpočet hledaných vlastností.

Přiložený program je naprogramovaný v jazyce C# kvůli jeho názornosti a jednoduchosti při práci s obrázky. Vytvořený modul dokáže zpracovat až 20 snímků za sekundu, což je možné považovat za zpracování v reálném čase a je pro naše potřeby dostačující. Ke zpomalení nedojde ani při větším počtu hráčů, než je na testovacích snímcích, jelikož většinu času zabírá předávání informací mezi jednotlivými částmi a samotné výpočty jsou rychlostně zanedbatelné. Další optimalizace by mohla zahrnovat použití rychlejšího jazyka, např. C++, který se více blíží nižším jazykům, které pracují na úrovní instrukcí procesoru. Také je zde možnost použití grafické karty pro paralelizované zpracování obrázku a poměrně velké zrychlení aplikace pro náročné operace.

Možná změna k lepšímu by mohla nastat také v našem značení robotů a hřiště. Ve světě se využívají složitější značení hráčů, která jsou však specifičtější a díky tomu se snadněji rozlišují například útočníci od obránců.

6 Reference

[1] FEDERATION, T. R. Objective of RoboCup. "<http://www.robocup.org/about-robocup/objective/>", 1998-2015.

[2] LIMITED, A. R. Aldebaran Robotics NAO V5 Evolution Humanoid Robot. "<http://www.active-robots.com/aldebaran-robotics-nao-evol-humanoid-robot>", 2015.

[3] BUDDEN, D. et al. Evaluation of Colour Models for Computer Vision Using Cluster Validation Techniques. In *RoboCup 2012: Robot Soccer World Cup XVI [papers from the 16th Annual RoboCup International Symposium, Mexico City, Mexico, June 18-24, 2012].*, s. 261–272, 2012. doi: 10.1007/978-3-642-39250-4˙24.

[4] BRUCE, J. – BALCH, M. T. a. V. Fast and inexpensive color image segmentation for interactive robots. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, 3, s. 2061–2066 vol.3, 2000. doi: 10.1109/IROS.2000.895274.

[5] ZICKLER, S. et al. SSL-Vision: The Shared Vision System for the RoboCup Small Size League. *RoboCup 2009: Robot Soccer World Cup XIII*. 2009, s. 425–436.

[6] PARKER, J. R. *Algorithms for Image Processing and Computer Vision*. Wiley Publishing, 2nd edition, 2010. ISBN 0470643854, 9780470643853.

[7] SOBEL, I. An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*. 2014.

[8] CANNY, J. A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* June 1986, 8, 6, s. 679–698. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851. Dostupné z: <http://dx.doi.org/10.1109/TPAMI.1986.4767851>.

[9] FELIPE PILON, B. K. D. Canny Edge Detector.

[10] SONKA, M. – HLAVAC, V. – BOYLE, R. *Image Processing, Analysis, and Machine Vision*. CENGAGE-Engineering, 3rd edition, 2007.

[11] DUDA, R. O. – HART, P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Commun. ACM*. January 1972, 15, 1, s. 11–15. ISSN 0001-0782. doi: 10.1145/361237.361242. Dostupné z: <http://doi.acm.org/10.1145/361237.361242>.

[12] CORPORATION, N. I. NI Vision 2011 Concepts Help. "<http://zone.ni.com/reference/en-XX/help/372916L-01/nivisionconcepts/edge_detection_concepts/>", 2011.

[13] LIMA, P. et al. Omni-directional catadioptric vision for soccer robots. *Robotics and Autonomous Systems*. 2001, 36, 2–3, s. 87 – 102. ISSN 0921-8890. Viewing, Sensing, Coordination and Learning in EuroRoboCup 2000.

[14] JONKER, P. – CAARLS, J. – BOKHOVE, W. Fast and Accurate Robot Vision for Vision Based Motion. 2001, 2019, s. 149–158.

[15] WASIK, Z. – SAFFIOTTI, A. Robust color segmentation for the RoboCup domain. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, 2, s. 651–654 vol.2, 2002. doi: 10.1109/ICPR.2002.1048386.

[16] PREETHA, M. – SURESH, L. – BOSCO, M. Image segmentation using seeded region growing. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, s. 576–583, March 2012. doi: 10.1109/ICCEET.2012.6203897.

[17] VINCENT, L. – SOILLE, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. Jun 1991, 13, 6, s. 583–598. ISSN 0162-8828. doi: 10.1109/34.87344.

[18] BLUM, H. A Transformation for Extracting New Descriptors of Shape. *Models for the Perception of Speech and Visual Form*. 1967, s. 362–380.

[19] ZHANG, T. Y. – SUEN, C. Y. A Fast Parallel Algorithm for Thinning Digital Patterns. *Commun. ACM*. March 1984, 27, 3, s. 236–239. ISSN 0001-0782. doi: 10.1145/357994.358023. Dostupné z: <http://doi.acm.org/10.1145/357994.358023>.

[20] CORPORATION, N. CUDA Parallel Computing Platform. "<http://www.nvidia.com/object/cuda_home_new.html>", 2015.

[21] FRANKLIN, W. R. Pnpoly-point inclusion in polygon test. "<http://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html>", 2006.